Jméno: ..... Test PDV 30. 5. 2025

(P1, 3b) U následujících tvrzení rozhodněte, zda platí:

- ☐ Atomické proměnné zajišťují, že mezi voláním několika atomických operací z jednoho vlákna nemůže žádné jiné vlákno do proměnné zapsat.
- ☐ Bitonické řazení lze použít jako metodu pro konstrukci řadicí sítě.
- ☐ Moderní procesory mají výrazně vyšší výpočetní výkon než moderní grafické karty.
- ☐ K "false sharing" může dojít nezávisle na velikosti řádky cache.
- ☐ Statické rozvrhování iterací `for` smyčky má vždy nižší overhead než dynamické rozvrhování.
- ☐ Direktiva `#pragma omp task` vytvoří nové vlákno, ve kterém se spustí obsah následujícího bloku.

(P2, 2b) Předpokládejte, že pomocí `#pragma omp for` paralelizujete for smyčku, kde každá iterace běží srovnatelně dlouho. Za jakých okolností může být vhodnější použít "guided" rozvrh (schedule) místo statického?

(P3, 2b) V jakých situacích je obecně vhodnější použít reader-writer lock místo obyčejného mutexu? V jaké situaci je naopak vhodnejší mutex? Zdůvodněte.

Jméno: ..... Test PDV 30. 5. 2025

(P4, 1b) Je následující kód korektní? Pokud ne, proč? Jak byste kód opravili?

```
class ConcurrentMap {
  std::unordered_map<std::string, int> _map{};
  std::mutex _map_lock{};

public:
  int get(const std::string& key) {
    _map_lock.lock();
    int value = _map.at(key);
    _map_lock.unlock();
    return value;
  }
    // ...
};
```

(P5, 2b) Uvažujme následující kód:

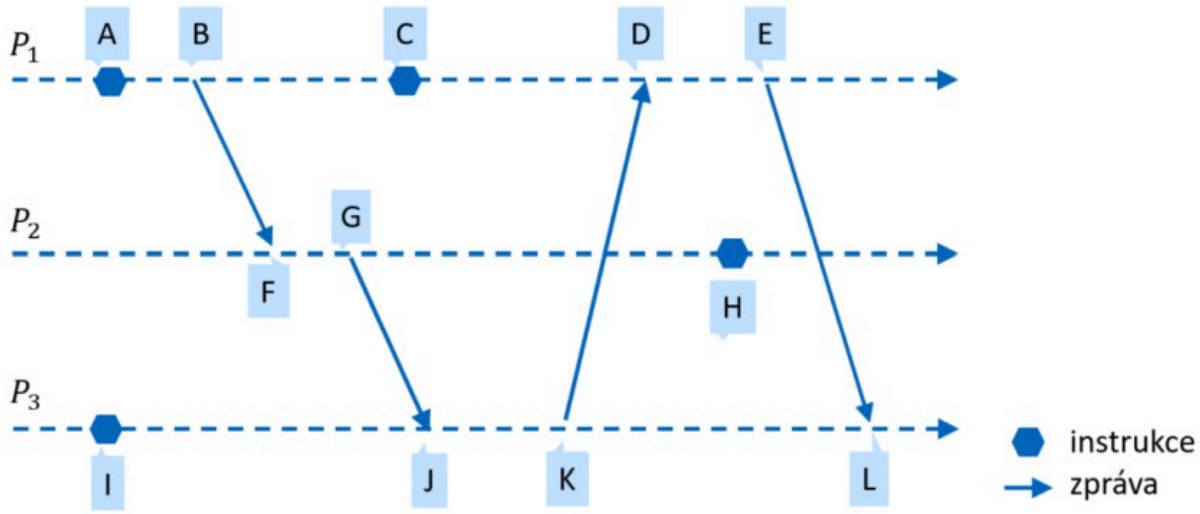
```
class LinkedStack {
    struct Node {int value; Node* next;};
    std::atomic<Node*> _head = nullptr;

public:
    void insert(int value) {
        Node* node = new Node{value, _head.load()};
        _head.compare_exchange_strong(node->next, node);
    }
    // ...
};
```

Která z následujících tvrzení platí?

- ☐ Při vkládání může docházet k úniku paměti (memory leak).
- ☐ Při vkládání může dojít k tomu, že ze seznamu se "ztratí" dříve vložená hodnota.
- ☐ Při vkládání může dojít k uváznutí (deadlock).
- ☐ Hodnota bude vždy úspěšně vložena do seznamu.

D1 (2b)

V distribuovaném systému vykonávajícím výpočet zachycený na obrázku běží *skalární* i *vektorové* logické hodiny. Na začátku výpočtu jsou všechny hodiny incializovány na *nulové* hodnoty.



Doplňte:

a. Hodnota skalárních hodin procesu P2 bezprostředně po události **H** bude ……..

b. Hodnota skalárních hodin procesu P3 bezprostředně po události **L** bude ……..

c. Hodnota vektorových hodin procesu P2 bezprostředně po události **G** bude ……..

d. Hodnota vektorových hodin procesu P3 bezprostředně po události L bude …….

D2 (3b)

Proces *P* používá Cristiánův algoritmus pro synchronizaci fyzických hodin. Proces *P* odeslal v lokálním čase **6h:40m:13.200s** požadavek na synchronizaci času k externím hodinám. V lokálním čase **6h:40m:13.320s** obdržel proces *P* od externích hodin zpět zprávu s časem externích hodin **6h:40m:13.230s**. Minimální latence komunikace od *P* k externím hodinám je **30ms** a minimální latence komunikace od externích hodin k procesu *P* je **10ms**.

Doplňte:

a. P1 si má přenastavit svůj lokální čas na hodnotu ……

b. Maximální odchylka času v procesu P1 a času externích hodin je ……

D3 (2b)

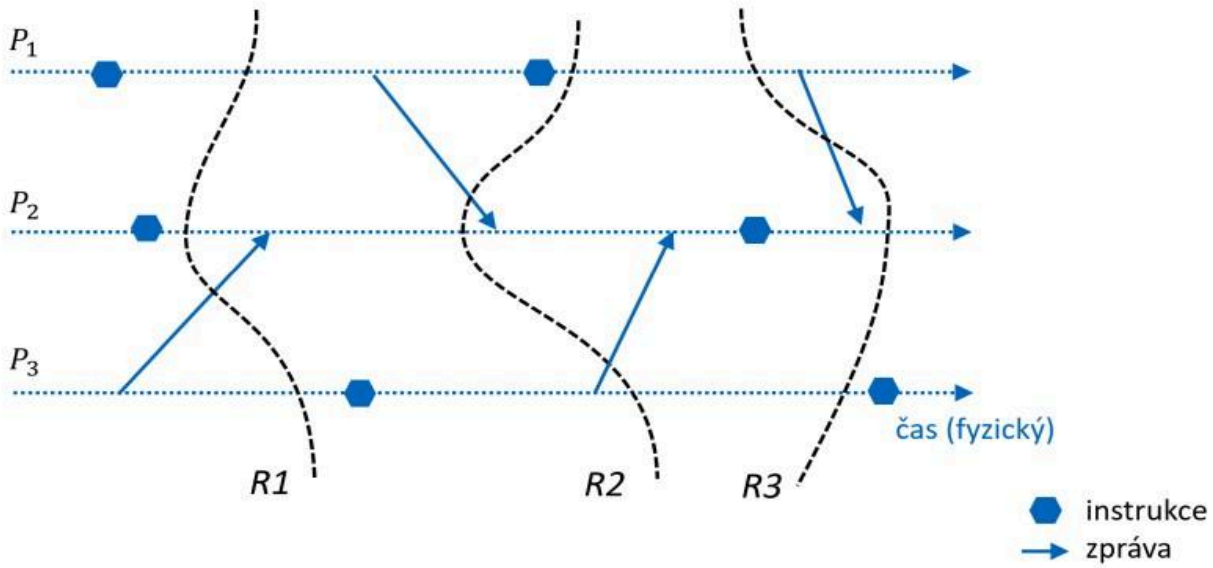
Označte všechna pravdivá tvrzení:

V částečně synchronním systému:

- ☐ Je doba doručení zpráv vždy shora omezená (danou maximální latencí).
- ☐ Může být doba doručení zpráv libovolně velká.
- ☐ Může být doba doručení zpráv nekonečně velká.
- ☐ Je doba doručení zpráv většinou shora omezená (danou maximální latencí).
- ☐ Se vyskytují dostatečně dlouhé časové intervaly, během kterých se systém chová jako synchronní systém.

D4 (2b)

Na obrázku jsou zachyceny tři řezy distribuovaného výpočtu.



Předpokládejme, že externí pozorovatel má v jakémkoliv fyzickém časovém okamžiku okamžitý přístup ke stavu všech procesů a všech kanálů.

Doplňte identifikátory řezů splňující následující tvrzení:

a. Řezy …… jsou konzistentní.

b. Řezy …… mohly být pozorovány externím pozorovatelem.

Jméno: ..... Test PDV 30. 5. 2025

D5 (2b)

Definujme **stupeň robustnosti** detektoru selhání jako maximální číslo N takové, že pokud v systému dosud neselhalo více než N libovolných procesů, tak je detektor stále úplný (a tedy že pokud selže N+1 procesů, tak už úplnost garantovat nelze).

Doplňte stupně robustnosti pro následující detektory běžící v distribuovaném systému sestávajícího z **10 procesů**:

a. Stupeň robustnosti *centralizovaného heartbeat* detektoru je: …….

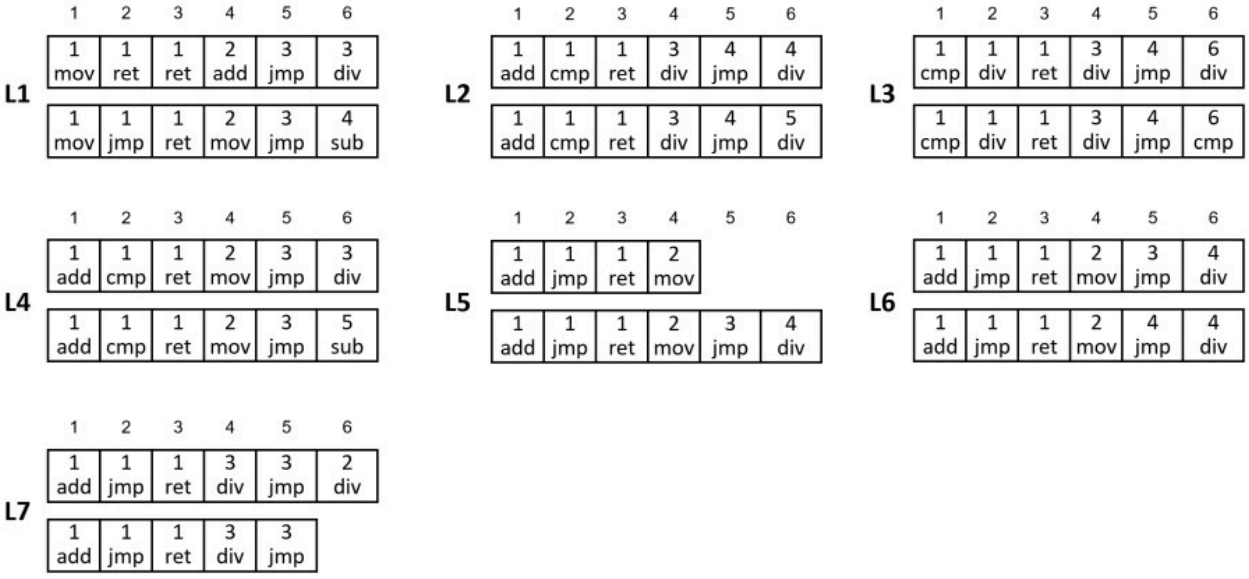
b. Stupeň robustnosti *oboustranného kruhového heartbeat* detektoru je …….

c. Stupeň robustnosti *all-to-all heartbeat* detektoru je …….

d. Stupeň robustnosti *SWIM* detektoru s K=3 je ……
(zde *K* označuje počet procesů, kterou jsou vybírány pro nepřímý ping-req.)

D6 (3b)

Zakroužkujte všechny dvojice logů, které se mohou vyskytnout v průběhu algoritmu RAFT:



D7 (2b)

Uvažujte centralizovaný algoritmus pro vyloučení procesů běžící v distribuovaném systému sestávajícím z 10 procesů (z nichž jeden je lídr).

Doplňte:

a. Pro vstup do kritické sekce je potřeba odeslat tento počet zpráv: ……..
b. Zpoždění klienta je tento počet komunikačních latencí: ……..
c. Synchronizační zpoždění je tento počet komunikačních latencí: …….

D8 (4b)

V distribuovaném systému sestávajícím z pěti procesů běží algoritmus **Bully** pro volbu lídra. Jako volební kritérium slouží identifikátor procesu (tj. proces **P5** má nejvyšší hodnotu volebního kritéria). V systému došlo k selhání dosavadního lídra **P5** a proces **P5** zůstává nadále nedostupný.

Předpokládejme, že selhání procesu **P5** detekuje nejdříve proces **P3**, který se chystá spustit volbu lídra pomocí algoritmu Bully. Dále předpokládejme, že v systému od tohoto okamžiku nebude docházet k dalším selháním (procesu ani kanálu).

Označte pravdivá tvrzení týkající se následného průběhu algoritmu Bully:

a. **P3** odešle zprávu **ELECTION**:
   - ☐ procesu **P1**
   - ☐ procesu **P4**
   - ☐ procesu **P2**

b. Ihned po přijetí zprávy **ELECTION** od procesu **P3**
   - ☐ proces **P1** pošle procesu **P4** zprávu **OK**
   - ☐ proces **P4** pošle procesu **P3** zprávu **OK**
   - ☐ proces **P2** pošle procesu **P1** zprávu **OK**
   - ☐ proces **P4** pošle procesu **P5** zprávu **ELECTION**
   - ☐ proces **P1** pošle procesu **P2** zprávu **ELECTION**
   - ☐ proces **P2** pošle procesu **P4** zprávu **ELECTION**

c. Po vypršení volebního time-out u procesu **P4** pošle proces **P4**
   - ☐ procesu **P1** zprávu **OK**
   - ☐ procesu **P3** zprávu **OK**
   - ☐ procesu **P1** zprávu **COORDINATOR(P4)**
   - ☐ procesu **P5** zprávu **COORDINATOR(P4)**
   - ☐ procesu **P2** zprávu **COORDINATOR(P4)**